SysCall Wrappers

CS3500 Course Project

Group 3

Week-2 Report

File I/O Wrappers

Yashvardhan, Kritang, Daksh

- Logger
  - **Logs messages** for various file operations (write, close) including **timestamps and details about the operation** (e.g., bytes written, file descriptor closed)
  - **Handles errors by logging** appropriate error messages when operations like closing a file descriptor fail.
- Safe_open
  - **Checks if the file exists** when the mode is "r" (read).
  - If the file does not exist, then **prints the first directory/file in the file path** that it could not access.
  - If the file exists, then attempts to open the file with the specified mode using fopen. If the file cannot be opened, it prints an error message and exits the program.
- Safe_read
  - Checks if the requested byte count is greater than the file size and handles errors during the read operation by printing appropriate error messages.
  - Attempts to **read a specified number of bytes** from a file descriptor into a buffer and **returns the number of bytes read** or an error code.

*All integrations to be done with master wrapper

Process Wrappers

Aditya Jain, Aditya Srivastava, Harsh Vardhan Daga, Shreyanshu Gurjar

- Process Monitoring and Logging
  - The wrapper outputs start time, end time, and elapsed time of the process.
  - It also keeps a log of **resource usage statistics** such as **CPU time, memory usage** and **context switches** and also the exit status of the process.
  - Helps detect abnormal process behavior, such as crashes or long execution times.

- Wait for all children
  - The wrapper calls waitpid() on all the child processes of the calling process present in /proc/<PID>/task/<TID>/children directory.
  - **Automatic Reaping:** Automates calling waitpid() on all child processes.
  - **Prevent Zombie Processes:** Ensures no child process is left in a zombie state.

- Custom Waitpid
  - This wrapper adds a **timeout feature** and handles errors during the child process state change.
  - Implements a 5-second timeout, printing a message if the child process isn't finished in time.
  - It also reports the **termination status** of the child process, detailing how it exited.

Process Wrappers

Aditya Jain, Aditya Srivastava, Harsh Vardhan Daga, Shreyanshu Gurjar

- Process Cloaking Wrapper
  - Fork Wrapper with dlsym:
    - **Intercepts fork()** using dlsym(RTLD_NEXT, "fork") to invoke the original function and execute custom actions in the child process.
  - Process Name Extraction:
    - **Extracts the child process name** from /proc/<pid>/stat and stores it in /home/reroot/Documents/hidden_process.txt.
  - Read Hooking to Hide Processes:
    - **Intercepts read() that are called by ps** and compare data with process names in hidden_process.txt. If a match is found, return 0 to hide the process.
  - File-Based Process Hiding:
    - Uses a text file (hidden_process.txt) to **track hidden process names** and filter them during read() calls. If a match is found, then those processes are not printed to the terminal.
  - How to Test:
    - Run the program, make sure to use the makefile to ensure library linking, check ps or top to ensure the **child process is hidden**, and verify hidden_process.txt for stored names.

Memory Wrappers

Raadhes, Mith, Dev, Rohan

- File tracker wrappers - This is under memory as forgetting to close files can cause memory leaks
  - **open_and_track**: Opens a file and stores the file descriptor in the file manager struct
  - **close_and_untrack**: Closes a file based on the file descriptor provided
  - **close_all_files**: Closes all files in the file manager

- SAFE MMAP:- This is a wrapper to safely execute the mmap instruction and give an overview analysis of the logging and tracking details for easier debugging.
  - **log_memory_usage**:- To output to the terminal all the dynamic allocations used in the memory-mapped files and libraries region of the virtual address space of the current process.
  - **safe_mmap:**- This function is the one that calls the mmap system called and based on the error that mmap returned with in case of failure, logs the details and informs of the exact error that occured.

Memory Wrappers

Raadhes, Mith, Dev, Rohan

- Aligned_Mmap :- This is a **wrapper for memory allocation** that ensures the returned memory address adheres to a specified alignment requirement for **optimal performance and compatibility**.
  - Reserves a memory region larger than the requested size to ensure alignment.
  - Computes the aligned memory address by rounding up the base address using bitwise operations.
  - **Stores the original base address** in the memory just before the aligned pointer. This is crucial for correctly deallocating memory later.

- Aligned_Munmap :- This is a **wrapper for deallocating memory** allocated by aligned_mmap, ensuring the correct base address is used to prevent errors.
  - **Retrieves the original base address** stored before the aligned pointer.
  - Calculates the total size of the memory region allocated during aligned_mmap.
  - Calls munmap with the original base address and the total size.